Exploring Security Commits in Python

Shiyu Sun[1], Shu Wang[1], Xinda Wang[1], Yunlong Xing[1], **Elisa Zhang**[2], Kun Sun[1]

[1]Center for Secure Information Systems, George Mason University
[2]Dougherty Valley High School

Outline

- Background
- Previous Solutions and Limitations
- Data Collection System
- Collected Data Analysis
- Security Patch Pattern Discovery

Background

- Python overtakes Java and C as the most popular programming language
- A large volume of OSS security patches (e.g., GitHub commits fixing vulnerabilities) are **silently released**.
  - Not reported to MITRE

```
From 7f9822a48213dd2feca845dbbb6bcb8beb9550de
Subject: [PATCH] Add blinding to a DSA signature

This is based on side channel attacks demonstrated by (NCC Group)
for ECDSA which are likely to be able to be applied to DSA.
```

  - Does not have explicit commit message

```
From 41bdc78544b8a93a9c6814b8bbbfef966272abbe
Subject: [PATCH] x86/tls: Validate TLS entries to protect espfix

Installing a 16-bit RW data segment into the GDT defeats espfix.
AFAICT this will not affect glibc, Wine, or dosemu at all.
```

- Timely security commit detection
- Assistance for auto-program repair tools

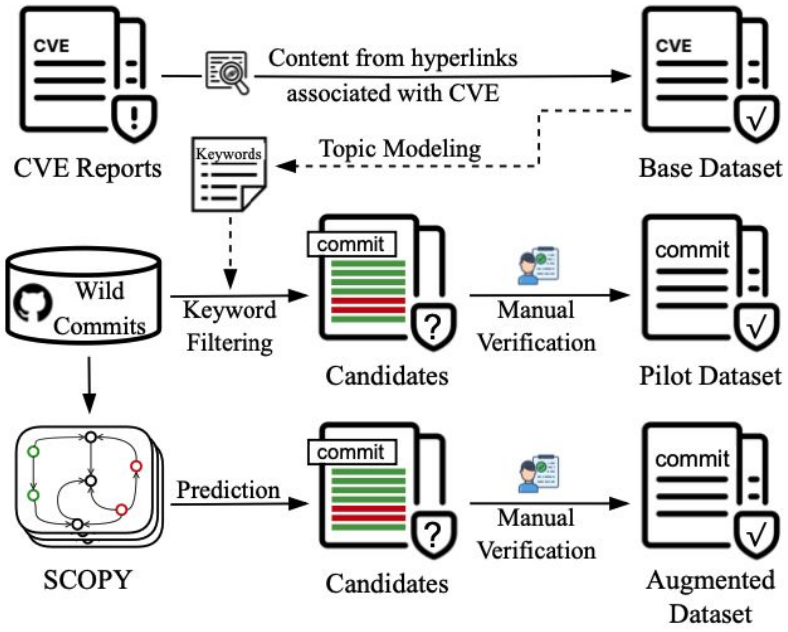
Previous Solutions and Limitations

- Commit Log [1]
  - Mining security keywords or security keyword sequence
    - Requiring well-maintained doc
- Source Code [2,3]
  - Mining security code feature or sequence
    - Missing importance structure semantics
- Our solution
  - Take both into consideration
    - Commit log: easy to mining
    - If no log, code: provide precise feature

```
From a2b169ffdef1d7c1755bade8138578423b35011b Mon Sep 17 00:00:00 2001
From: Alexander Todorov <atodorov@otb.bg>
Date: Mon, 7 Nov 2022 17:52:57 +0200
Subject: [PATCH] Clean HTML input when generating history diff

helps us prevent XSS attacks

 tcms/core/history.py | 9 +++++++++
 1 file changed, 9 insertions(+)

diff --git a/tcms/core/history.py b/tcms/core/history.py
index abc2edc264..76a9fcccc2 100644
--- a/tcms/core/history.py
+++ b/tcms/core/history.py
@@ -8,6 +8,8 @@
 from simple_history.admin import SimpleHistoryAdmin
 from simple_history.models import HistoricalRecords

+from tcms.core.templatetags.extra_filters import bleach_input
+

 def diff_objects(old_instance, new_instance, fields):
     """
@@ -20,6 +22,13 @@ def diff_objects(old_instance, new_instance, fields):
         field_diff = []
         old_value = getattr(old_instance, field.attname)
         new_value = getattr(new_instance, field.attname)
+
+        # clean stored XSS
+        if isinstance(old_value, str):
+            old_value = bleach_input(old_value)
+        if isinstance(new_value, str):
+            new_value = bleach_input(new_value)
+
```

[1] Zhou, Yaqin, et al. "Spi: Automated identification of security patches via commits." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31.1 (2021): 1-27.
[2] Wang, Xinda, et al. "Patchdb: A large-scale security patch dataset." *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2021.
[3] Wang, Shu, et al. "GraphSPD: Graph-based security patch detection with enriched code semantics." *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023.

Our Solution: A Comprehensive Security Patch Collection System



① Base dataset: From MITRE

CVE®

② Pilot dataset: Analyze Commit Msg

```
Subject: [PATCH] fix: reject NUL character as path element

See: https://github.com/Pylons/pyramid/security/advisories/GHSA-j
```

③ Augmented dataset: Analyze Code

```
-_seps = {'/', os.sep}
+_invalid_element_chars = {'/', os.sep, '\x00'}
```

① Base Dataset: Extracting Security Patches from MITRE

For each CVE entry, we download its patch from the Git Hyperlink and exclude the commits that are not written in Python or only focus on security-unrelated modifications (e.g., renaming and refactoring).

CVE®

Full-Screen View

CVE-ID

CVE-2023-24816

Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

IPython (Interactive Python) is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language. Versions prior to 8.1.0 are subject to a command injection vulnerability with very specific prerequisites. This vulnerability requires that the function `IPython.utils.terminal.set_term_title` be called on Windows in a Python environment where ctypes is not available. The dependency on `ctypes` in `IPython.utils._process_win32` prevents the vulnerable code from ever being reached in the ipython binary. However, as a library that could be used by another tool `set_term_title` could be called and hence introduce a vulnerability. Should an attacker get untrusted input to an instance of this function they would be able to inject shell commands as current process and limited to the scope of the current process. Users of ipython as a library are advised to upgrade. Users unable to upgrade should ensure that any calls to the `IPython.utils.terminal.set_term_title` function are done with trusted or filtered input.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC:https://github.com/ipython/ipython/blob/3f0bf05f072a91b2a3042d23ce250e5e906183fd/IPython/utils/terminal.py#L103-L117
- URL:https://github.com/ipython/ipython/blob/3f0bf05f072a91b2a3042d23ce250e5e906183fd/IPython/utils/terminal.py#L103-L117
- MISC:https://github.com/ipython/ipython/blob/56e6925dfa50e2c7f4a6471547b8176275db7c25/IPython/utils/_process_win32.py#L20
- URL:https://github.com/ipython/ipython/blob/56e6925dfa50e2c7f4a6471547b8176275db7c25/IPython/utils/_process_win32.py#L20
- MISC:https://github.com/ipython/ipython/commit/385d69325319a5972ee9b5983638e3617f21cb1f
- URL:https://github.com/ipython/ipython/commit/385d69325319a5972ee9b5983638e3617f21cb1f
- MISC:https://github.com/ipython/ipython/security/advisories/GHSA-29gw-9793-fvw7
- URL:https://github.com/ipython/ipython/security/advisories/GHSA-29gw-9793-fvw7

```
From 991849c247fc208628879e7ca2923b3c218a5a75 Mon Sep 17 00:00:00 2001
From: Konstantin Weddige <konstantin.weddige@lutrasecurity.com>
Date: Sat, 3 Dec 2022 19:14:09 +0100
Subject: [PATCH] Fix CVE-2023-24816 by removing legacy code.

Remove legacy code that might trigger a CVE.

Currently set_term_title is only called with (semi-)trusted input that
contain the current working directory of the current IPython session. If
an attacker can control directory names, and manage to get a user cd
into this directory the attacker can execute arbitrary commands
contained in the folder names.
```

CVE-2023-24816 Detail

② Pilot Dataset: Augmenting via Keyword-filtering

Rationale: 8% GitHub commits are security patches without a CVE-ID; only 46% of indexed CVE records contain the corresponding security fixes.

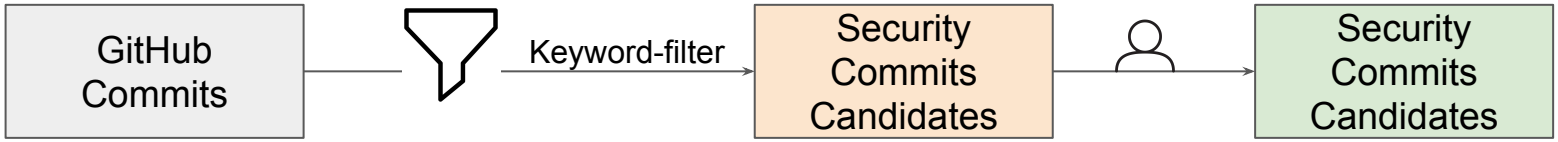
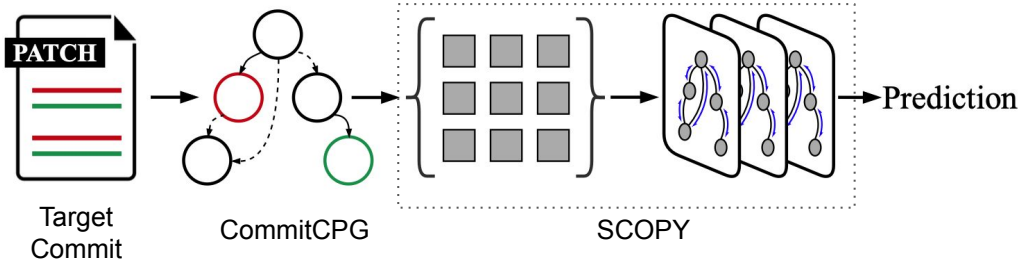


TABLE I: Security-related keywords for commit filtering.

#Tokens	Keywords
1-gram	attack, bypass, CVE, DoS, exploit, injection, leakage, malicious, overflow, smuggling, spoofing, unauthorized, underflow, vulnerability
2-gram	access control, open redirect, race condition
3-gram	denial of service, out of bound, dot dot slash

③ Augmented Dataset: Extending via Graph Learning

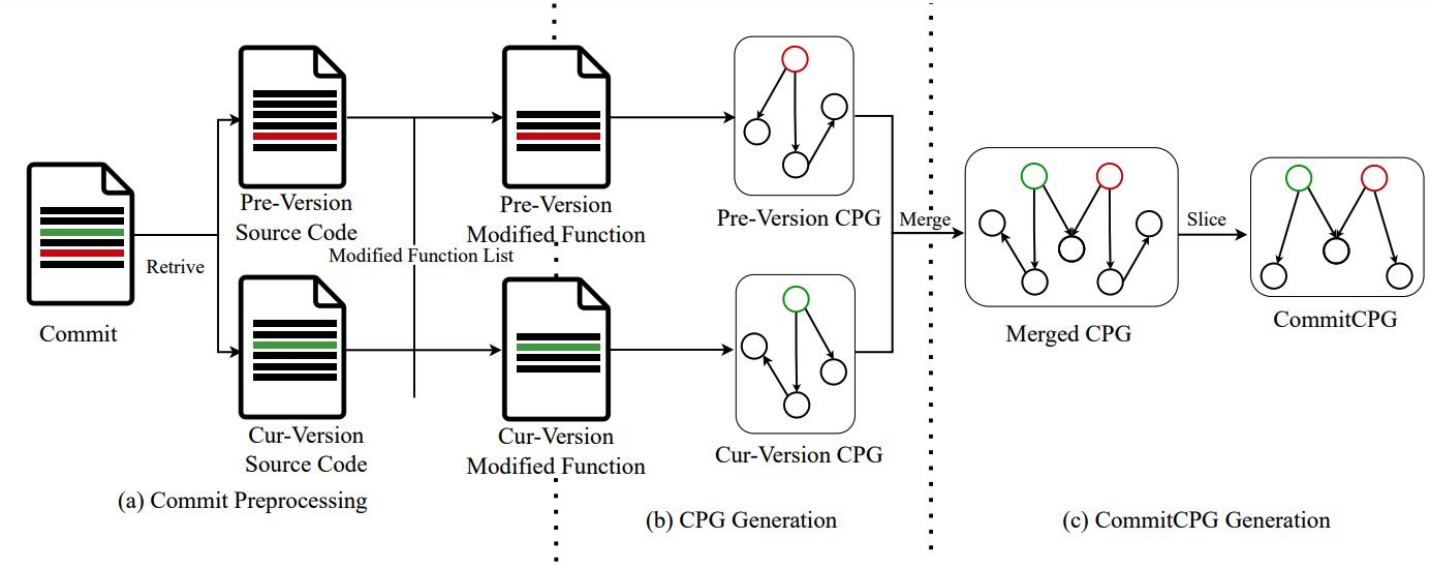
The pilot dataset overlooks the commits that lack security keywords in the commit messages, while these commits may provide additional variants in syntax and semantics



- **CommitCPG**: graph representation of inherent code change structures.
- **SCOPY**: graph learning of structural and sequential semantics for security commit detection.

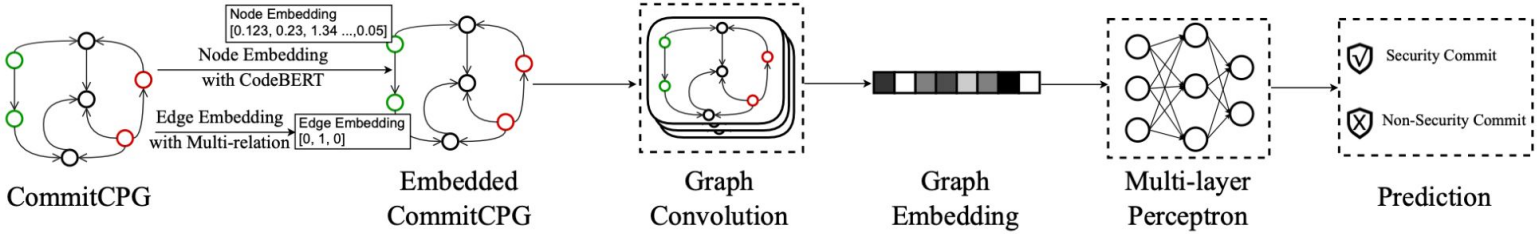
CommitCPG: From Commit to Graph

- Challenge: how to construct CommitCPG?
  - Syntax and semantics: program dependency graph
  - Changes and relations: merged and sliced previous-version and current version commit graph



SCOPY: Detect Security Patches from CommitCPGs

- Challenges and Solutions:
  - How to **embed** the CommitCPG?
    - Node: **CodeBERT;** Edge: **[AST, CFG, DDG];** Graph: **Graph Neural Network**
  - How to learn **multiple attributes** of divergent program dependencies from previous-version commit and current-version commit?
    - **Graph Convolution with Multi-Head Attention**



Implementation & Evaluation

Implementation

- 6K new LoC in Scala and Python on top of Joern parser and PyTorch library.

Research Questions

- **RQ1**: Can the graph learning-based method improve the data collection efficiency?
- **RQ2**: How various and representative are the collected security commits?
- **RQ3**: What are the unique patterns of security commits in Python?

Dataset Construction (RQ1)

The composition of PySecDB

- 1,258 security commits
- 2,791 non-security commits

Commit \ Dataset	Base	Pilot	Augmented	Total
Security	729	400	129	1258
Non-Security	2134	535	122	2791

Efficiency of keyword filtering and SCOPY

- Keyword filtering: improve **30%**+ efficiency
- SCOPY: improve **40%**+ efficiency

Method	# Candidates	# Verified SC*	Ratio
Random [5]	-	-	6-10%
Keywords	935	400	42.70%
SCOPY	251	129	51.39%

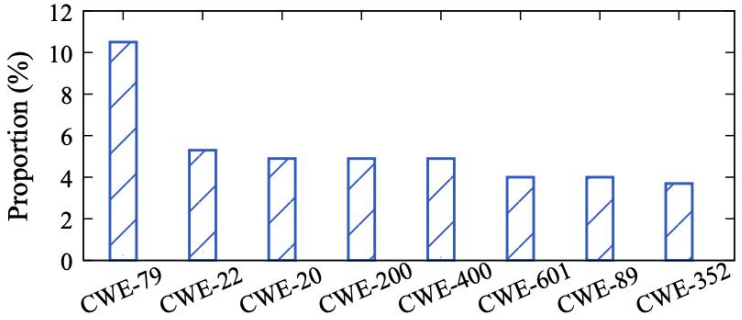
* SC = Security Commits.

Security Commits Categorization and Distribution (RQ2)

Top 5 repositories by number of security commits

Repository	#SecurityCommits	Proportion
django	166	13.20%
twisted	87	6.91%
glance	54	4.29%
pillow	41	3.26%
numpy	39	3.10%
Total of Top 5	**387**	**30.76%**

The top 8 CWE types in PySecDB



Patch Patterns (RQ3)

Pattern	#Commits	Proportion
1) Add or Update Sanity Checks	416	37.12%
2) Update API Usage	241	19.16%
3) Update Regular Expressions	189	15.02%
4) Restrict Security Properties	183	14.55%
5) Others	178	14.15%
Total	**1258**	**100%**

Patch Patterns (RQ3)

Add or Update Sanity Check

Usage scenarios: authentication property verification, access control, HTTP request checking

```
commit c658b4f3e57258acf5f6207a90c2f2169698ae22
diff --git a/core.py b/core.py
@@ -112,7 +112,7 @@ def actualsys() :
    if attemps == 6:
        ## Brute force protection
        raise Exception("Too many password attempts.")
-   if os.environ.get('GITHUB_ACTIONS') != "":
+   if os.environ.get('GITHUB_ACTIONS') == "true":
        logging.warning("Running on Github Actions")
        actualsys()
    elif uname == cred.name and pwdhash == cred.pass:
```

An example of security commit that fixes an authorization bypass exploit vulnerability (CVE-2022-46179)

Patch Patterns (RQ3)

Update API Usage

Usage scenarios: OS command injection, code injection, and regular expression injection

```
1  commit f6753a1a1c63fade6ad418fbda827c6750ab0bda
2  diff --git a/weblate/trans/forms.py b/weblate/trans/
       forms.py
3  @@ -37,6 +37,7 @@
4  ...
5 +from django.utils.html import escape
6  ...
7 -    label = str(unit.translation.language)
8 +    label = escape(unit.translation.language)
9  ...
```

An example of security commit that fixes an XSS vulnerability (CVE-2022-24710)

Patch Patterns (RQ3)

Update Regular Expressions

Usage scenarios: avoid XSS, SQL injection, and open redirect vulnerabilities

```
1  commit fc2c1ea1b8d795094abb15ac73cab90830534e04
2  diff --git a/.../model.py b/.../model.py
3  @@ -772,13 +772,13 @@ def _get_filter(self):
4  if self.queueid:
5  -     ... = '%s'" % (self.queueid)
6  +     ... = '%s'" % (re.sub("[\"']", "", self.queueid))
```

An example of security commit that fixes a SQL injection vulnerability (CVE-2014-125082)

Patch Patterns (RQ3)

Restrict Security Properties

Usage scenarios: updating boolean flags from True to False or vice versa, adding more arguments to methods, or adding security decorators.

```
1  commit 60a3fe559c453bc36b0ec3e5dd39c1303640a59a
2  diff --git a/src/nsupdate/settings/base.py b/src/
       nsupdate/settings/base.py
3  @@ -283,7 +283,7 @@
4  ...
5 -CSRF_COOKIE_HTTPONLY = False
6 +CSRF_COOKIE_HTTPONLY = True
7  ...
```

An example of security commit that fixes a vulnerability where the sensitive cookie does not have a ‘HttpOnly’ flag (CVE-2019-25091)

Conclusion and Future Work

- We publicize a large-scale Python security commit dataset named PySecDB
- We leverage the commit message and source code change to capture the security attributes of each commit
- We conduct a large-scale empirical study of security commits by analyzing PySecDB of 119 CWE categories across 351 repositories
- The Register interviews the paper, sparking discussion.
- Our Data Collection System can be applied to other languages. We will extend the dataset with other popular languages in future work.

Thank you!

Contacts: elisaz.ca@gmail.com, ssun20@gmu.edu

Dataset: https://github.com/SunLab-GMU/PySecDB